IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **DeWitt, Jr. et al.**	§	
	§	Group Art Unit: **2181**
Serial No. **10/757,269**	§	
	§	Examiner: **Lai, Vincent**
Filed: **January 14, 2004**	§	
	§	
For: **Method and Apparatus for Autonomically Initiating Measurement of Secondary Metrics Based on Hardware Counter Values for Primary Metrics**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on December 14, 2006.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-29.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE.
2. Claims withdrawn from consideration but not canceled: NONE.
3. Claims pending: 1-29.
4. Claims allowed: NONE.
5. Claims rejected: 1-29.
6. Claims objected to: NONE.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-29.

STATUS OF AMENDMENTS

There are no amendments after the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1:

The present invention provides a method in a data processing system for processing instructions. (Specification, page 87, line 2, to page 90, line 17) The present invention determines whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention enables counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention determines if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention enables counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

Independent claim 25:

The present invention provides for a computer program product in a recordable-type computer readable medium for processing instructions. (Specification, page 125, line 23, to page 126, line 11; and Specification, page 87, line 2, to page 90, line 17) The present invention provides first instructions for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the

indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention provides second instructions for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention provides third instructions for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention provides fourth instructions for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

A person having ordinary skill in the art would be able to derive computer instructions on a computer readable medium as recited in claim 15, as well as dependent claims 16-28, given **Figures 29** and **39**, performing the steps described in the specification at page 66, line 21, to page 68, line 19; and page 90, line 18, to page 92, line 9, without undue experimentation.

Independent claim 15:

The present invention provides for an apparatus for processing instructions. (Specification, page 23, lines 12-21; page 26, lines 10-13; page 28, line 19, to page 29, line 9; and Specification, page 87, line 2, to page 90, line 17) The present invention provides means for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention provides means for enabling counting, by the processor, of each first event associated with a primary metric of the

execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention provides means for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention provides means for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

The apparatus recited in claim 29 may be an apparatus comprised of functional units such as execution units **220**, **222**, **224**, **226**, **228**, and **230** in processor **210** of **Figure 2**. The processor and functional units providing means for determining and means for enabling to perform the steps described in the specification at page 66, line 21, to page 68, line 19; and page 90, line 18, to page 92, line 9, without undue experimentation.

Dependent claim 3 and 17:

The present invention stores the indicator in a performance instrumentation shadow cache. (Specification, page 33, lines 9-16) The present invention checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions. (Specification, page 33, lines 17-22)

Dependent claim 5 and 19:

The present invention provides that the indicator is a separate instruction. (Specification, page 33, lines 9-16)

Dependent claim 9 and 23:

The present invention provides that the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters. (Specification, page 82, line 29, to page 83, line 6)

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION (Claims 1-29)

Whether claims 1-29 are obvious under 35 U.S.C. § 103(a) over Yates et al. (U.S. Patent No. 6,549,959 B1) in view of Holmberg (U.S. Patent No. 6,233,679 B1).

ARGUMENT

A. GROUND OF REJECTION, Claims 1-29

A.1. Group A: Claims 1-29

Claim 1 is representative of the claims in this group and reads as follows:

1. A method in a data processing system for processing instructions, the method comprising:

responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (emphasis added)

Yates and Holmberg, taken alone or in combination, fail to teach or suggest enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter; determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor

autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

Yates is directed to a central processing unit that is programmed to execute first and second processes. The first process is programmed to generate a second representation in a computer memory of information of the second process stored in the memory in a first representation. A main memory is divided into pages for management by a virtual memory manager that uses a table stored in the memory. Direct Memory Access (DMA) monitoring circuitry and/or software is designed to monitor DMA memory write transactions to a main memory of a computer by a DMA device of the computer. The DMA detects when the first representation is overwritten by a DMA memory write transaction initiated by the second process, without the second process informing the first process of the DMA memory write transaction. (see Yates, Abstract)

The Examiner alleges that Yates teaches enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter in the following section:

> Hot spot detector **122** recognizes addresses that frequently recur in a set of profile packets. Once a hot spot is recognized, the surrounding entries in the profile may indicate (by physical address) a region of code that is frequently executed in correlation with the recurring address, and the path through the physical pages. Hot spot detector **122** conveys this information to TAXi translator 124, which in turn translates the binary.

(Yates, column 55, lines 53-60)

In this section, Yates describes recognizing addresses that frequently recur in a set of profile packets. In detecting the recurring addresses, Yates describes the following:

> In its most-common mode of operation, profiler **400** awaits a two-part trigger signal (**516**, **522** of **FIG. 5a**) to start sampling events, and then *records every profileable event **416** in a dense sequence, including every profileable event that occurs*, until it stops (for instance, on exhaustion of the buffer into which profile information is being collected), *as opposed to a conventional profiler that records every n^{th} event, or records a single event every n microseconds*. The profile information records both the source and destination addresses of most control flow transfers. Entries describing individual events are collected into the

machine's general register file, and then stored in a block as a profile packet. This blocking of events reduces memory access traffic and exception overhead. (*emphasis added*)

(Yates, column 54, line 59, to column 55, line 5)

In this section, Yates teaches counting every event based on a hot spot. In contradistinction, the present invention counts each **first event** associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction. Thus, Yates teaches away from the present invention by counting every event and determining a reoccurrence of events. Yates does not teach or suggest enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter.

Furthermore, the Examiner alleges that Yates teaches determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value in column 55, lines 54-58, shown above. As discussed above, Yates describes recognizing addresses that frequently recur in a set of profile packets, and Yates teaches counting every event based on a hot spot. Consequently, Yates teaches away from the present invention by counting every packet. Furthermore, Yates does not teach or suggest determining if the **count of the first events** associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value.

The Examiner acknowledges that Yates does not teach or suggest enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. However, the Examiner alleges that Holmberg teaches this feature in the following section:

In **FIG. 2** the hardware used in the unit **121** for collecting statistics regarding if a branch is taken or not, is shown. Thus, for collecting statistics regarding a certain conditional branch instruction in the program memory, the address of that instruction is placed in a register **201**, here termed Measured Address Register (MAR). This address is compared in a block **203** with the

instruction address currently pointed to by the program counter and which is available in a block **205**.

The two addresses are compared in the block **203** and if the two addresses are identical a first counter in a block **211** is incremented by one. The output from the block **203** is also fed to an AND block **207**. To the AND block **207**, a signal indicating if the branch was taken or not is also fed. Thus, the output from the block **207** increments a second counter **209** each time the branch in the instruction in the memory address register is taken.

In general, two out of the following statistics counts needs to be collected for setting the branch prediction bits:

the number of times the conditional branch is taken
the number of times the conditional branch is not taken
the total number of times the conditional branch instruction is executed.

(Holmberg, column 4, line 47, to column 5, line 2)

In this section, Holmberg uses two different counters to count two of: the number of times the conditional branch is taken, the number of times the conditional branch is not taken, or the total number of times the conditional branch instruction is executed. Thus, once Holmberg encounters the first conditional jump instruction, it is loaded into a measured address register. Then, Holmberg increments the counter each time the program from which statistics are collected executes the conditional branch instruction associated with the address stored in the MAR and when the corresponding branch is taken. Thus, Holmberg counts all of the times the conditional jump instruction is taken and the second event. Therefore, Holmberg does not teach or suggest enabling counting, by the processor, of **each second event** associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Since the references fail to teach or suggest responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit; enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the

indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter; determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter, the Examiner has failed to establish a *prima facie* case of obviousness, because the Examiner does not show where each and every claim limitation is taught or fairly suggested by the applied prior art.

The applied references do not teach or suggest each and every claim limitation; therefore, Yates and Holmberg, taken alone or in combination, do not render claim 1 obvious. Independent claims 15 and 29 recite similar subject matter addressed above with respect to claim 1 and are allowable for similar reasons. Since claims 2-14 and 16-28 depend from claims 1 and 15, the same distinctions between Yates and Holmberg and the invention recited in claims 1, 15, and 29 apply for these claims. Additionally, claims 2-14 and 16-28 recite other additional combinations of features not taught or suggested by the references.

Furthermore, no suggestion is present in any of the references to modify the references to include such features. That is, there is no teaching or suggestion in Yates and Holmberg that a problem exists for which enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter; determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second

hardware counter, is a solution. To the contrary, Yates appears to teach counting every event based on a hot spot and Holmberg appears to teach counting all of the times the conditional jump instruction is taken.

Moreover, neither reference teaches or suggests the desirability of incorporating the subject matter of the other reference. That is, there is no motivation offered in either reference for the alleged combination. The Examiner alleges that the motivation would be "utilizing a second counter would allow greater ability to track and thus better predictions can be made." (See Final Office Action dated October 10, 2006, page 4) The present invention provides for counting of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction and counting of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction. As discussed above, Yates appears to teach counting every event based on a hot spot and Holmberg appears to teach counting all of the times the conditional jump instruction is taken. Neither reference teaches or suggests counting each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction and counting each second event associated with a secondary metric of the execution of a portion of code associated with the instruction. Thus, the only teaching or suggestion to even attempt the alleged combination is based on a prior knowledge of Appellants' claimed invention, thereby constituting impermissible hindsight reconstruction using Appellants' own disclosure as a guide.

One of ordinary skill in the art, being presented only with Yates and Holmberg, and without having a prior knowledge of Appellants' claimed invention, would not have found it obvious to combine and modify Yates and Holmberg to arrive at Appellants' claimed invention, as recited in claim 1. To the contrary, even if one were somehow motivated to combine Yates and Holmberg, and it were somehow possible to combine the systems, the result would not be the invention, as recited in claim 1. The resulting system would still fail to count each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, and fail to count each second event associated with a secondary metric of the execution of a portion of code associated with the instruction.

In view of the above, Appellants respectfully submit that Yates and Holmberg, taken alone or in combination, fail to teach or suggest the features of claims 1, 15, and 29. At least by

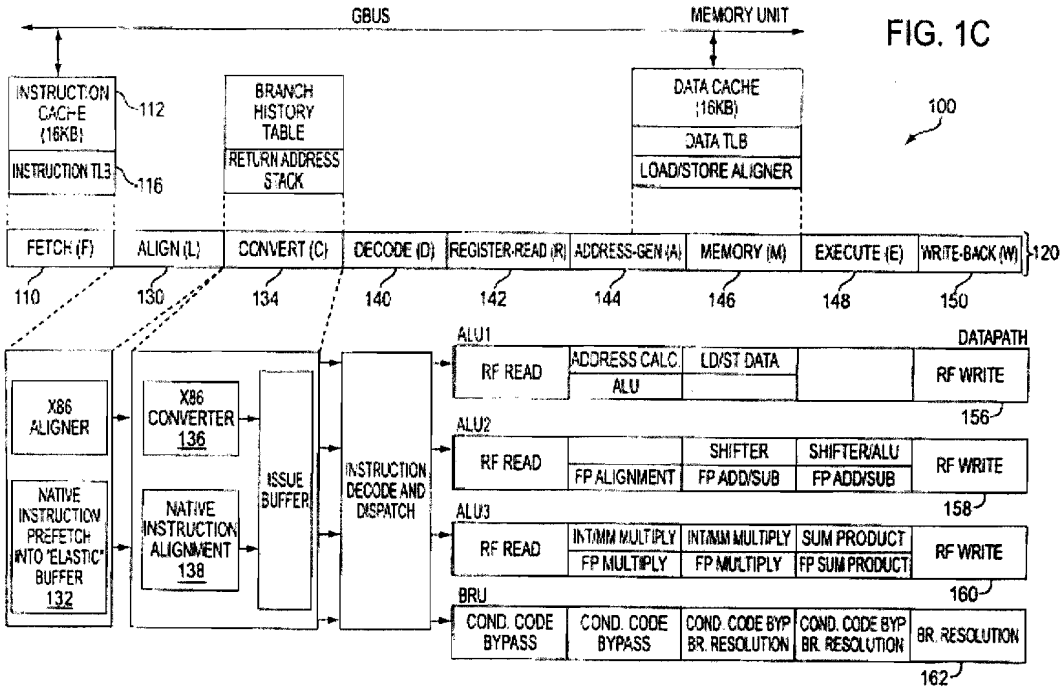
virtue of their dependency on claims 1, 15, and 29, the features of dependent claims 2-14 and 16-28 are not taught or suggested by Yates and Holmberg, whether taken individually or in combination. Accordingly, Appellants respectfully request the rejection of claims 1-29 under 35 U.S.C. § 103 not be sustained.

A.2. Group B: Claims 3 and 17

Claim 3 is representative of the claims in this group and reads as follows:

> 3. The method of claim 1, wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

Appellants respectfully submit that Yates and Holmberg, taken alone or in combination, do not teach or suggest wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions. The Examiner alleges that Yates teaches these features in the following Figure and section:



(Yates, Figure 1c)

While Yates illustrates an issue buffer, Yates fails to describe an issue buffer anywhere within the reference. Thus, one of ordinary skill in the art would assume the issue buffer illustrated by Yates to be a buffer that issues instructions. In contradistinction, the present invention stores the indicator in a performance instrumentation shadow cache, which is a separate area of storage from the instruction buffer.

> A transition from X86 code to Tapestry code (for instance, a successful probe exception, see section VI, infra) may be an abort **550**, **552** event. Profiler **400** is configured to allow the choice between entirely discarding the aborted packet or padding out and then spilling the partial packet to the ring buffer before abort **550**, **552** occurs. This choice is implemented in the code of the X86-to-Tapestry transition handler **320**. **FIG. 5b** is a block diagram of a portion of profiler **400**, the logic **554** to collect and format a profile entry **430**, **440** into a processor register.

(Yates, column 67, lines 36-45)

In this section, Yates describes a profiler, which is counting every event based on a hot spot, may be configured to allow the choice between entirely discarding the aborted packet or padding out and then spilling the partial packet to the ring buffer before abort. Nowhere does Yates describe checking the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

In view of the above, Yates and Holmberg, taken either alone or in combination, fail to teach or suggest the specific features recited in independent claims 1 and 15, from which claims 3 and 17 depend. Accordingly, Appellants respectfully request the rejection of claims 3 and 17 under 35 U.S.C. § 103 not be sustained.

A.3. Group C: Claims 5 and 19

Claim 5 is representative of the claims in this group and reads as follows:

5. The method of claim 1, wherein the indicator is a separate instruction.

Appellants respectfully submit that Yates and Holmberg, taken alone or in combination, do not teach or suggest wherein the indicator is a separate instruction. The Examiner alleges that Yates teaches this feature in the following figure:

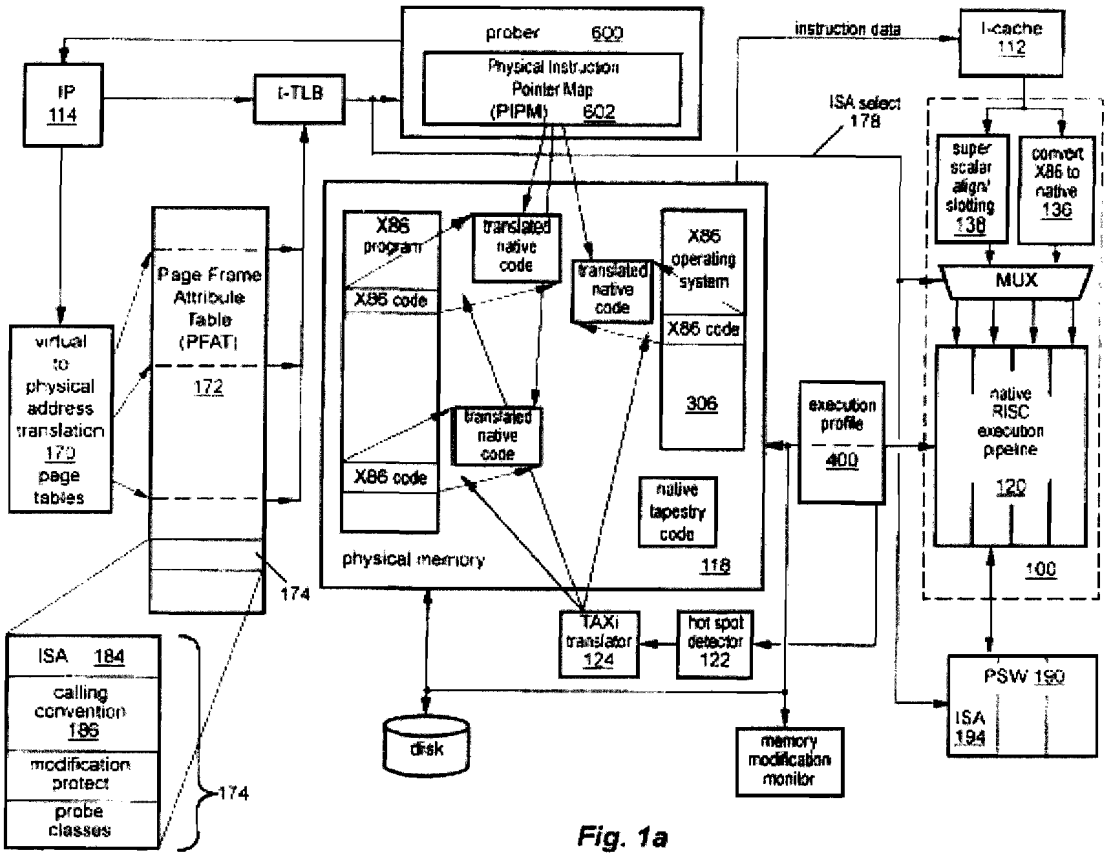


Fig. 1a

(Yates, Figure 1a)

In this Figure and the related text, Yates describes a tapestry processor that fetches instructions from an instruction cache, or from memory, from a location specified by instruction pointer, with virtual-to-physical address translation provided by I-TLB. The instructions fetched from the I-cache are executed by a RISC execution pipeline. Furthermore, Yates teaches that the memory regions have associated first and second indicator elements, the indicator elements each having a value indicating the architecture or data storage convention under which instructions from the associated region are to be executed. Thus, Yates teaches that the indicators are associated with memory regions, and Yates fails to teach or suggest an indicator that is a separate instruction.

In view of the above, Yates and Holmberg, taken either alone or in combination, fail to teach or suggest the specific features recited in independent claims 1 and 15, from which claims 5 and 19 depend. Accordingly, Appellants respectfully request the rejection of claims 5 and 19 under 35 U.S.C. § 103 not be sustained.

A.4. Group D: Claims 9 and 23

Claim 9 is representative of the claims in this group and reads as follows:

> 9. The method of claim 1, wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

Appellants respectfully submit that Yates and Holmberg, taken alone or in combination, does not teach or suggest wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters. The Examiner alleges that this feature is taught by Yates in column 55, lines 54-58, shown above, and by Holmberg at column 4, line 64 to column 5, line 2, shown above. Yates teaches counting every event based on a hot spot. Holmberg teaches the use of two different counters to count two of:

- the number of times the conditional branch is taken,
- the number of times the conditional branch is not taken, or
- the total number of times the conditional branch instruction is executed.

Thus, Holmberg counts either

- the number of times the branch is taken **and** the number of times the branch in not taken,
- the number of time the branch is taken **and** the total number of times the conditional branch instruction is executed, or
- the number of times the branch is not taken **and** the total number of times the conditional branch instruction is executed.

There is no teaching or suggestion in Holmberg that the number of times the branch is taken and the number of times the branch is not taken are combined to create the total number of times the conditional branch instruction is executed. Moreover, Holmberg only provides for two counters so that a comparison may be made between the counters. Thus, Yates and Holmberg, taken alone or in combination, fail to teach or suggest wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

In view of the above, Yates and Holmberg, taken either alone or in combination, fail to teach or suggest the specific features recited in independent claims 1 and 15, from which claims

9 and 23 depend. Accordingly, Appellants respectfully request the rejection of claims 9 and 23 under 35 U.S.C. § 103 not be sustained.

CONCLUSION

In view of the above, Appellants respectfully submit that claims 1-29 are allowable over the cited prior art and that the application is in condition for allowance. Accordingly, Appellants respectfully request the Board of Patent Appeals and Interferences to reverse the rejections set forth in the Final Office Action.

Respectfully submitted,

/Francis Lammes/
Francis Lammes
Reg. No. 55,353
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method in a data processing system for processing instructions, the method comprising:

responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

2. The method of claim 1, wherein the instruction is received in an instruction cache in the processor.

3. The method of claim 1, wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

4. The method of claim 1, wherein the instruction is received in a bundle by an instruction cache in the processor and wherein the indicator comprises at least one spare bit in a field in the bundle.

5. The method of claim 1, wherein the indicator is a separate instruction.

6. The method of claim 1, wherein the first events include at least one of an entry into a module, an exit from a module, an entry into a subroutine, an exit from a subroutine, an entry into a function, an exit from a function, a start of input/output, a completion of input/output, and the execution of the instruction.

7. The method of claim 1, wherein determining whether an indicator is associated with the instruction comprises:

determining, by an instruction cache, whether the indicator is present in a field within the instruction.

8. The method of claim 1, wherein the enabling the counting of first events includes sending a first signal to the performance monitor unit, wherein the performance monitor unit counts each first event associated with execution of the instruction using the first hardware counter, and

wherein enabling the counting of second events includes sending a second signal to the performance monitor unit, wherein the performance monitor unit counts each second event associated with execution of a portion of code associated with the instruction using the second hardware counter.

9. The method of claim 1, wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

10. The method of claim 1, wherein enabling counting, by the processor, of each second event associated with the secondary metric of the execution of the portion of code associated with the instruction includes:

generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value; and

sending the interrupt to an interrupt handler of a performance monitoring application, wherein the interrupt handler of the performance monitoring application initiates counting of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction.

11. The method of claim 10, wherein the interrupt handler instruments other instructions in the portion of code associated with the instruction to include the indicator.

12. The method of claim 10, wherein the interrupt handler initiates the second hardware counter and associates the second hardware counter with the portion of code.

13. The method of claim 1, wherein the portion of code associated with the instruction includes at least one of instructions of a same class of instructions as the instruction and instructions within a same method or routine as the instruction.

14. The method of claim 1, wherein the first metric is different from the second metric.

15. A computer program product in a recordable-type computer readable medium for processing instructions comprising:

first instructions responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

second instructions for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

third instructions for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

fourth instructions for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events

associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

16. The computer program product of claim 15, wherein the instruction is received in an instruction cache in the processor.

17. The computer program product of claim 15, wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

18. The computer program product of claim 15, wherein the instruction is received in a bundle by an instruction cache in the processor and wherein the indicator comprises at least one spare bit in a field in the bundle.

19. The computer program product of claim 15, wherein the indicator is a separate instruction.

20. The computer program product of claim 15, wherein the first events include at least one of an entry into a module, an exit from a module, an entry into a subroutine, an exit from a subroutine, an entry into a function, an exit from a function, a start of input/output, a completion of input/output, and the execution of the instruction.

21. The computer program product of claim 15, wherein the first instructions include:

instructions for determining, by an instruction cache, whether the indicator is present in a field within the instruction.

22. The computer program product of claim 15, wherein the second instructions for enabling the counting of first events include instructions for sending a first signal to the performance monitor unit, wherein the performance monitor unit counts each first event associated with execution of the instruction using the first hardware counter, and wherein enabling the counting of second events includes sending a second signal to the performance monitor unit, wherein the performance monitor unit counts each second event associated with execution of a portion of code associated with the instruction using the second hardware counter.

23. The computer program product of claim 15, wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

24. The computer program product of claim 15, wherein the fourth instructions for enabling counting, by the processor, of each second event associated with the secondary metric of the execution of the portion of code associated with the instruction include:

instructions for generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value; and

instructions for sending the interrupt to an interrupt handler of a performance monitoring application, wherein the interrupt handler of the performance monitoring application initiates

counting of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction.

25. The computer program product of claim 24, wherein the interrupt handler instruments other instructions in the portion of code associated with the instruction to include the indicator.

26. The computer program product of claim 24, wherein the interrupt handler initiates the second hardware counter and associates the second hardware counter with the portion of code.

27. The computer program product of claim 15, wherein the portion of code associated with the instruction includes at least one of instructions of a same class of instructions as the instruction and instructions within a same method or routine as the instruction.

28. The computer program product of claim 15, wherein the first metric is different from the second metric.

29. An apparatus for processing instructions, the apparatus comprising:

means for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

means for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the

instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

means for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

means for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.